Here we will learn how to use sysmon to hunt for Mimkatz, and this is a lab where we will setup the Sysmon in a way to log events of Mimikatz:

First we will configure sysmon installation to capture logs that we can use to hunt for MimiKatz:

As we know if we use /? With the sysmon executable we will get the help of sysmon:

Command: sysmon64.exe /?

Here we will use these flags:

-i: to install the Sysmon driver and service

-acceptula: this will accept the ula of installation like if we don't a window will pop up which will ask us to agree to terms of use

-h md5, sha254,iphash : here we will specify the hashing algorithm that will be used to hash the log files for its integrity and we specify all the supported hashes.

-l : it will log the load of modules

-n : its to log the network connections

```
c:\Tools\Sysmon>Sysmon64.exe -i -accepteula -h md5,sha256,imphash -l -n

System Monitor v6.10 - System activity monitor
Copyright (C) 2014-2017 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Sysmon installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting Sysmon..
Sysmon started.
```

Lets look at the sysmon configuration with the -c flag:

```
c:\Tools\Sysmon>Sysmon.exe -c

System Monitor v6.10 - System activity monitor
Copyright (C) 2014-2017 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Current configuration:
  - Service name:                   Sysmon
  - Driver name:                    SysmonDrv
  - HashingAlgorithms:              MD5,SHA256,IMPHASH
  - Network connection:             enabled
  - Image loading:                  enabled
  - CRL checking:                   disabled
  - Process Access:                 disabled

No rules installed
```

We see the hashing algorithm, the image loading and the Network connections are enabled, and we see there is no rule configured.

Now we will add our Config file, but lets look at it first and see how it looks like:

```
<Sysmon schemaversion="3.40">
   <!-- Capture all hashes -->
   <HashAlgorithms>*</HashAlgorithms>

   <EventFiltering>
      <!-- Event ID 1 == Process Creation. -->
      <ProcessCreate onmatch="include"/>

      <!-- Event ID 2 == File Creation Time. -->
      <FileCreateTime onmatch="include"/>

      <!-- Event ID 3 == Network Connection. -->
      <NetworkConnect onmatch="include"/>

      <!-- Event ID 5 == Process Terminated. -->
      <ProcessTerminate onmatch="include"/>

      <!-- Event ID 6 == Driver Loaded.-->
      <DriverLoad onmatch="include"/>

      <!-- Event ID 7 == Image Loaded. -->
      <ImageLoad onmatch="include"/>

      <!-- Event ID 8 == CreateRemoteThread. -->
      <CreateRemoteThread onmatch="include"/>

      <!-- Event ID 9 == RawAccessRead. -->
      <RawAccessRead onmatch="include"/>

      <!-- Event ID 10 == ProcessAccess. -->
      <ProcessAccess onmatch="include"/>

      <!-- Event ID 11 == FileCreate. -->
      <FileCreate onmatch="include"/>

      <!-- Event ID 12,13,14 == RegObject added/deleted, RegValue Set, RegObject Renamed. -->
      <RegistryEvent onmatch="include"/>

      <!-- Event ID 15 == FileStream Created. -->
      <FileCreateStreamHash onmatch="include"/>

      <!-- Event ID 17 == PipeEvent. -->
      <PipeEvent onmatch="include"/>
   </EventFiltering>
</Sysmon>
```

Here we see the onmatch=”include” that mean if the event matches then it will log it, if its exclude then if it matched it wouldn’t log that event, but here its include on all the rules. The first line of each rule is comment here but what matter is the second line of the rule like if see in the first rule in the second line we see “<ProcessCreate onmatch=”include”>” , here the “ProcessCreate” is the event of new process created and then we have the action which is “include” mean to log it, mean to log process creation.

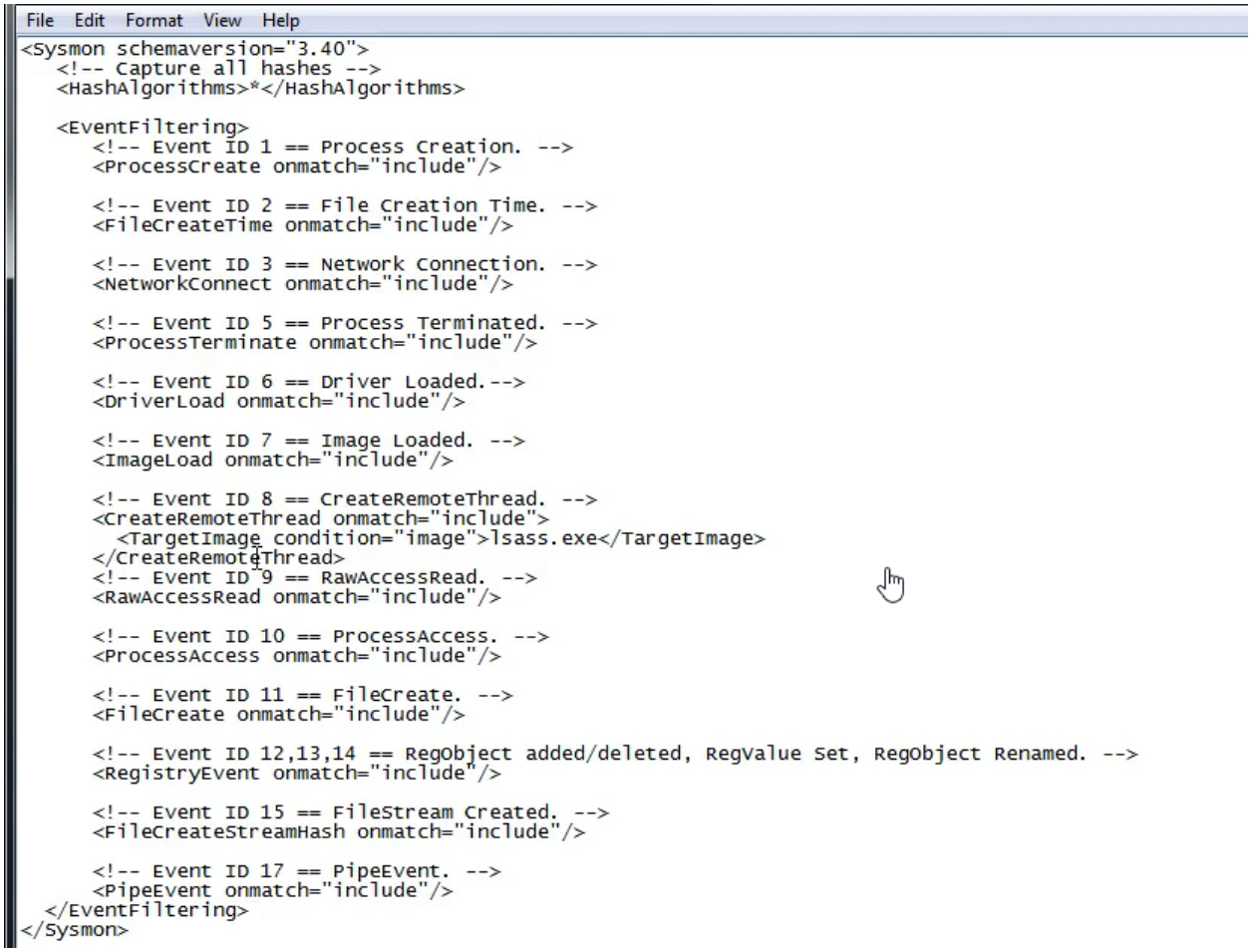
As we know the Mimkatz is a tool that dump the credentials from the memory of the lsass.exe by injecting it self to the process (we know it access the process not injecting into it, here we act like that so we learn more and better), so lets go to the rule “CreateRemoteThread” and add a trigger condition for injection on lsass.exe process, mean any process that inject it self to the lsass.exe by creating a new thread which that’s how the injection work where it creates a new thread in that process, so any process that inject into lsass.exe , its event will be logged.

```
<!-- Event ID 8 == CreateRemoteThread. -->
<CreateRemoteThread onmatch="include">
  <TargetImage condition="image">lsass.exe</TargetImage>
</CreateRemoteThread>
```

It’s the same as the html, here the end of the rule/tag is when we see the tag ends with </tag>.

Here we added <TargetImage> tag bellow the rule as its indented, It means it belongs to the top, the target image mean that if there is a new thread created in image lsass.exe log it.

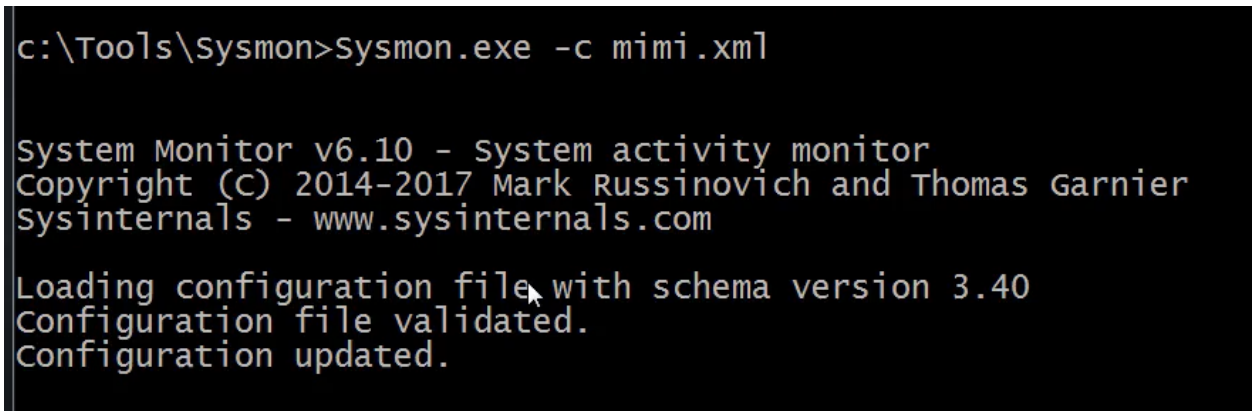
Here how the configuration file looks like:



```
File  Edit  Format  View  Help
<Sysmon schemaversion="3.40">
   <!-- Capture all hashes -->
   <HashAlgorithms>*</HashAlgorithms>

   <EventFiltering>
      <!-- Event ID 1 == Process Creation. -->
      <ProcessCreate onmatch="include"/>

      <!-- Event ID 2 == File Creation Time. -->
      <FileCreateTime onmatch="include"/>

      <!-- Event ID 3 == Network Connection. -->
      <NetworkConnect onmatch="include"/>

      <!-- Event ID 5 == Process Terminated. -->
      <ProcessTerminate onmatch="include"/>

      <!-- Event ID 6 == Driver Loaded.-->
      <DriverLoad onmatch="include"/>

      <!-- Event ID 7 == Image Loaded. -->
      <ImageLoad onmatch="include"/>

      <!-- Event ID 8 == CreateRemoteThread. -->
      <CreateRemoteThread onmatch="include">
        <TargetImage condition="image">lsass.exe</TargetImage>
      </CreateRemoteThread>
      <!-- Event ID 9 == RawAccessRead. -->
      <RawAccessRead onmatch="include"/>

      <!-- Event ID 10 == ProcessAccess. -->
      <ProcessAccess onmatch="include"/>

      <!-- Event ID 11 == FileCreate. -->
      <FileCreate onmatch="include"/>

      <!-- Event ID 12,13,14 == RegObject added/deleted, RegValue Set, RegObject Renamed. -->
      <RegistryEvent onmatch="include"/>

      <!-- Event ID 15 == FileStream Created. -->
      <FileCreateStreamHash onmatch="include"/>

      <!-- Event ID 17 == PipeEvent. -->
      <PipeEvent onmatch="include"/>
   </EventFiltering>
</Sysmon>
```

Lets save it and then add to the sysmon with -c flag:

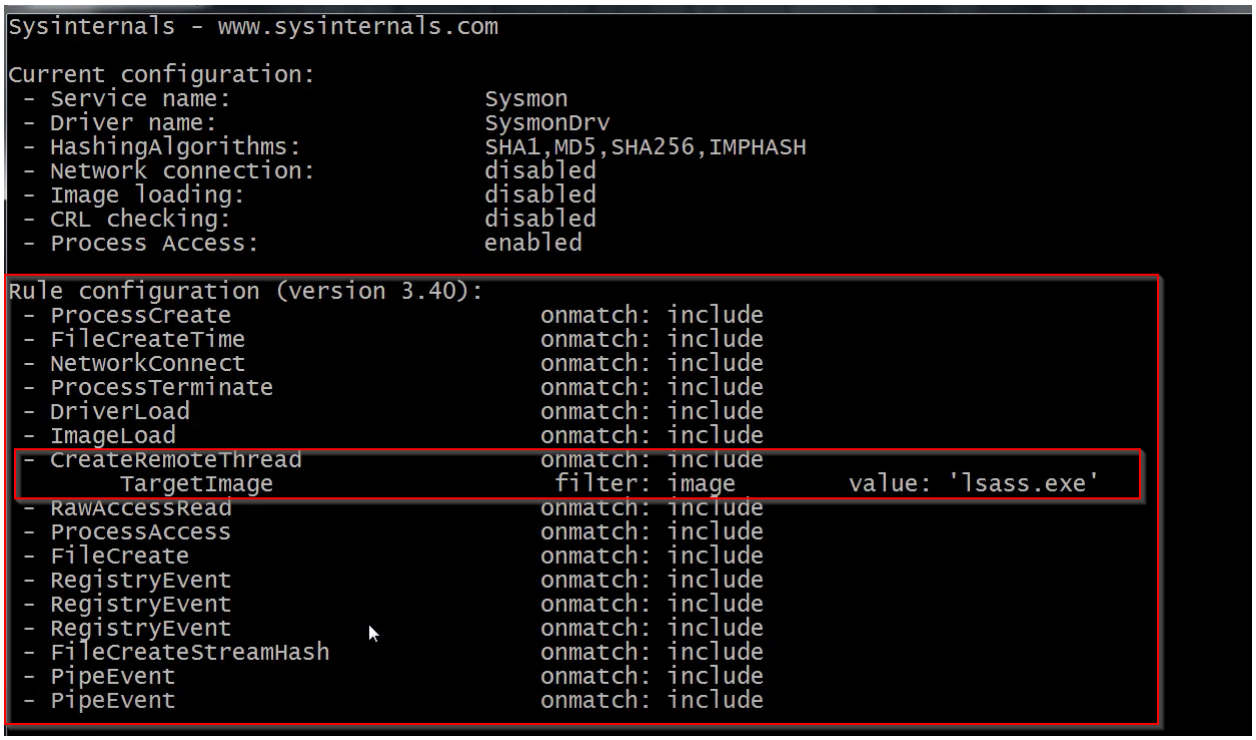
Command: sysmon64.exe -c NewCOnfigurationFile.xml



```
c:\Tools\Sysmon>Sysmon.exe -c mimi.xml


System Monitor v6.10 - System activity monitor
Copyright (C) 2014-2017 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Loading configuration file with schema version 3.40
Configuration file validated.
Configuration updated.
```

Lets check the configuration to see if it changed with -c flag:

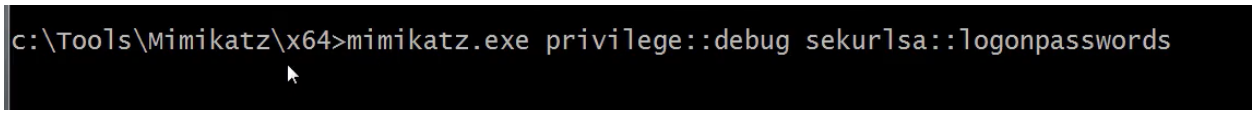
Command: sysmon64.exe -c



We see that the configuration file is added and see the lsass.exe filter:

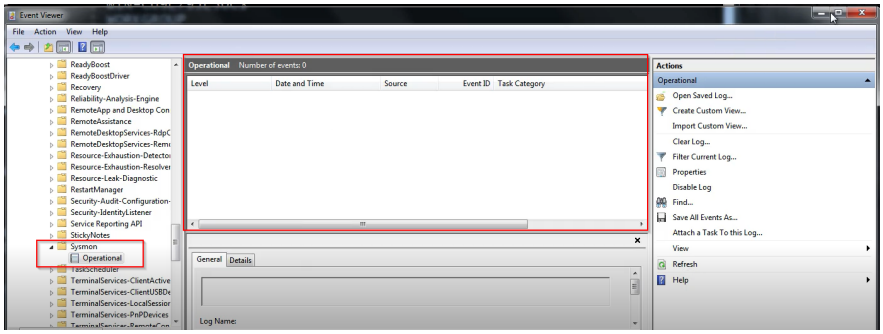
Here we will now execute the memekatz and then analyze the captured logs/events.

Running or executing mimikatz:



When we run this command, it will create a thread within the lsass.exe process so it can dump the credentials from it (no it will not inject it self into the process of lsass.exe, mimikatz will access the process not inject into it. we are just trying to learn so).

We run the command so lets check the event Viewer to see if it created any log:



We see nothing its because the mimkatz dosnt inject it self into the lsass.exe process to dump the credentials, as we know injecting is to get control of the process but mimkatz dosnt need to inject it self to the lsass.exe process but rather access it and read it without to inject it self.

So we need to make changes to the “ProcessAccess” event of sysmon which is Event ID 10, to log the process access:

So we will add that condition that we added for the “CreateRemoteThread” to the “ProcessAccess” event.

```
<!-- Event ID 10 == ProcessAccess. -->
<ProcessAccess onmatch="include">
  <TargetImage condition="image">lsass.exe</TargetImage>
</ProcessAccess>
```

So this mean where there is a process access happen and the targeted image is lsass.exe then log it as that’s what the mimkatz do.

and that’s how the configuration file look like:

```
File  Edit  Format  View  Help
<Sysmon schemaversion="3.40">
   <!-- Capture all hashes -->
   <HashAlgorithms>*</HashAlgorithms>

   <EventFiltering>
      <!-- Event ID 1 == Process Creation. -->
      <ProcessCreate onmatch="include"/>

      <!-- Event ID 2 == File Creation Time. -->
      <FileCreateTime onmatch="include"/>

      <!-- Event ID 3 == Network Connection. -->
      <NetworkConnect onmatch="include"/>

      <!-- Event ID 5 == Process Terminated. -->
      <ProcessTerminate onmatch="include"/>

      <!-- Event ID 6 == Driver Loaded.-->
      <DriverLoad onmatch="include"/>

      <!-- Event ID 7 == Image Loaded. -->
      <ImageLoad onmatch="include"/>

      <!-- Event ID 8 == CreateRemoteThread. -->
      <CreateRemoteThread onmatch="include">
        <TargetImage condition="image">lsass.exe</TargetImage>
      </CreateRemoteThread>

      <!-- Event ID 9 == RawAccessRead. -->
      <RawAccessRead onmatch="include"/>

      <!-- Event ID 10 == ProcessAccess. -->
      <ProcessAccess onmatch="include">
        <TargetImage condition="image">lsass.exe</TargetImage>
      </ProcessAccess>

      <!-- Event ID 11 == FileCreate. -->
      <FileCreate onmatch="include"/>

      <!-- Event ID 12,13,14 == RegObject added/deleted, RegValue Set, RegObject Renamed. -->
      <RegistryEvent onmatch="include"/>

      <!-- Event ID 15 == FileStream Created. -->
      <FileCreateStreamHash onmatch="include"/>

      <!-- Event ID 17 == PipeEvent. -->
      <PipeEvent onmatch="include"/>
   </EventFiltering>
</Sysmon>
```

Save it and then load it back into the Sysmon so it capture or log the event where a process accesses the lsass.exe.

Loading:

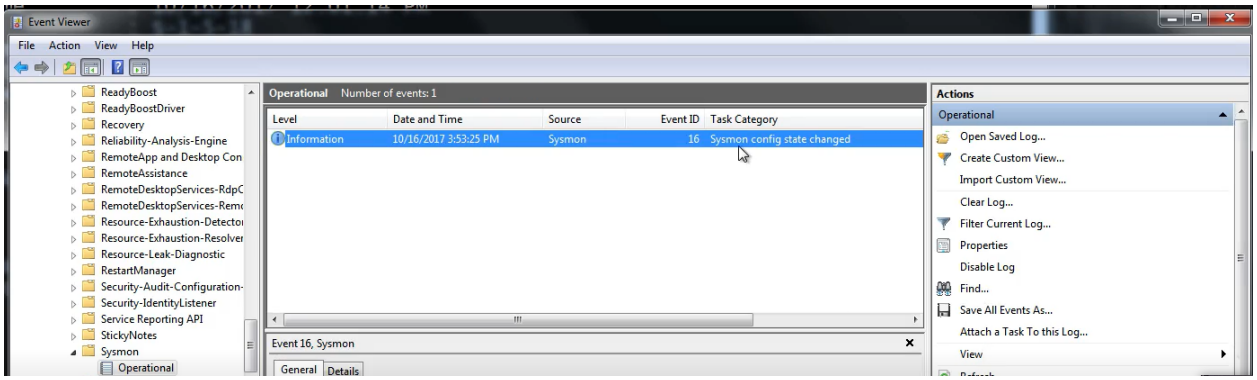
```
c:\Tools\Sysmon>Sysmon.exe -c mimi.xml


System Monitor v6.10 - System activity monitor
Copyright (C) 2014-2017 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Loading configuration file with schema version 3.40
Configuration file validated.
Configuration updated.
```

We didn’t get any error, lets check the configuration:

```
Current configuration:
 - Service name:                  Sysmon
 - Driver name:                   SysmonDrv
 - HashingAlgorithms:             SHA1,MD5,SHA256,IMPHASH
 - Network connection:            disabled
 - Image loading:                 disabled
 - CRL checking:                  disabled
 - Process Access:                enabled

Rule configuration (version 3.40):
 - ProcessCreate                        onmatch: include
 - FileCreateTime                       onmatch: include
 - NetworkConnect                       onmatch: include
 - ProcessTerminate                     onmatch: include
 - DriverLoad                           onmatch: include
 - ImageLoad                            onmatch: include
 - CreateRemoteThread                   onmatch: include
       TargetImage                       filter: image        value: 'lsass.exe'
 - RawAccessRead                        onmatch: include
 - ProcessAccess                        onmatch: include
       TargetImage                       filter: image        value: 'lsass.exe'
 - FileCreate                           onmatch: include
 - RegistryEvent                        onmatch: include
 - RegistryEvent                        onmatch: include
 - RegistryEvent                        onmatch: include
 - FileCreateStreamHash                 onmatch: include
 - PipeEvent                            onmatch: include
 - PipeEvent                            onmatch: include
```

Here we see its added:

When the configuration file is updated, it will create an Event ID 16, lets first check this and then execute the mimkatz:



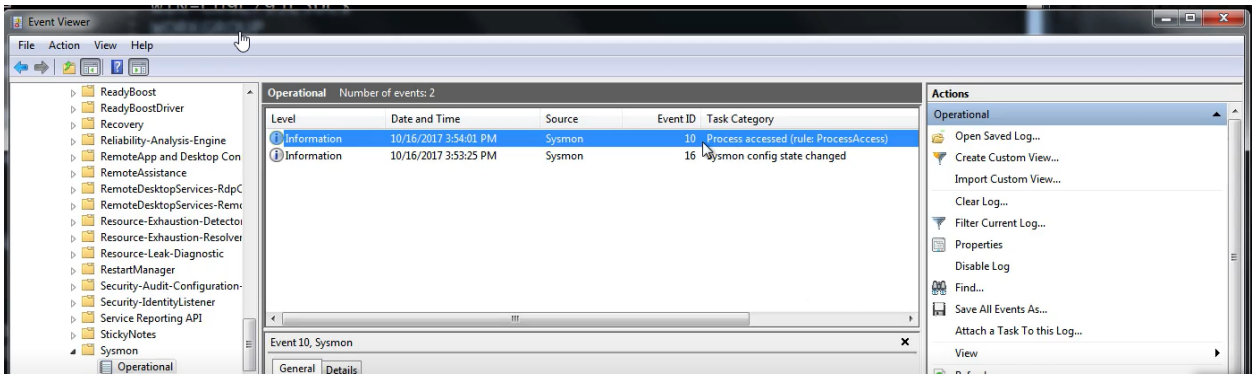
Here we see it, it logged the event of new loaded configuration to the sysmon:

We will execute mimakatz again and see if this time any events/logs will be created:

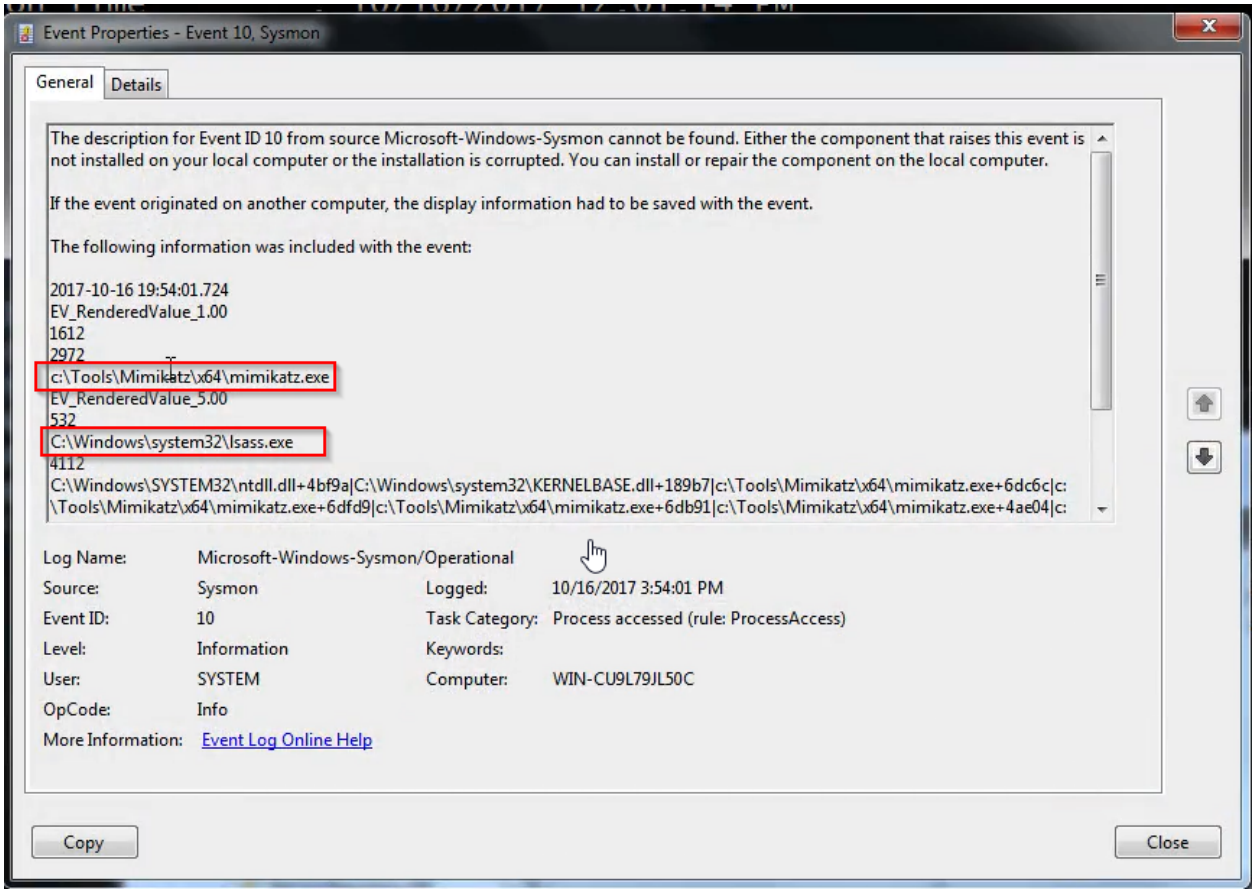
```
c:\Tools\Mimikatz\x64>mimikatz.exe privilege::debug sekurlsa::logonpasswords
```

Lets check the Event Viewer for any events if created:

Again the sysmon logs are located in event viewer in “In “Application And services > Microsoft > Windows > sysmon > operational”:



We got hit on ID 10 which is process access Event ID, we will double click the Event or that log to see its details:



Here we see that the image or the process that accessed which image or process which we see the mimkatz.exe access the process of lsass.exe.

We can see the same info in the PowerShell

First lets just get all the sysmon logs without filtering:

```
PS C:\Windows\system32> Get-WinEvent -FilterHashtable @{logname="Microsoft-Windows-Sysmon/Operational"}


   ProviderName: Microsoft-Windows-Sysmon

TimeCreated                     Id LevelDisplayName Message
-----------                     -- ---------------- -------
10/16/2017 3:54:01 PM           10 Information      Process accessed:...
10/16/2017 3:53:25 PM           16 Information      Sysmon config state changed:...
```

Here we have the -FilterHashtable, it's a way of filter for logs and we see the logname which here specify which logs we want

As we saw In the Event Viewer 2 events, here we also see 2 events which one is logged when sysmon configuration was changed mean we loaded a new configuration, so as we changed the configuration file and loaded it again so its logged with event ID 16 and we also see the Event ID 10 which is the Process access:

Lets add the filter to get the Events with ID 10:

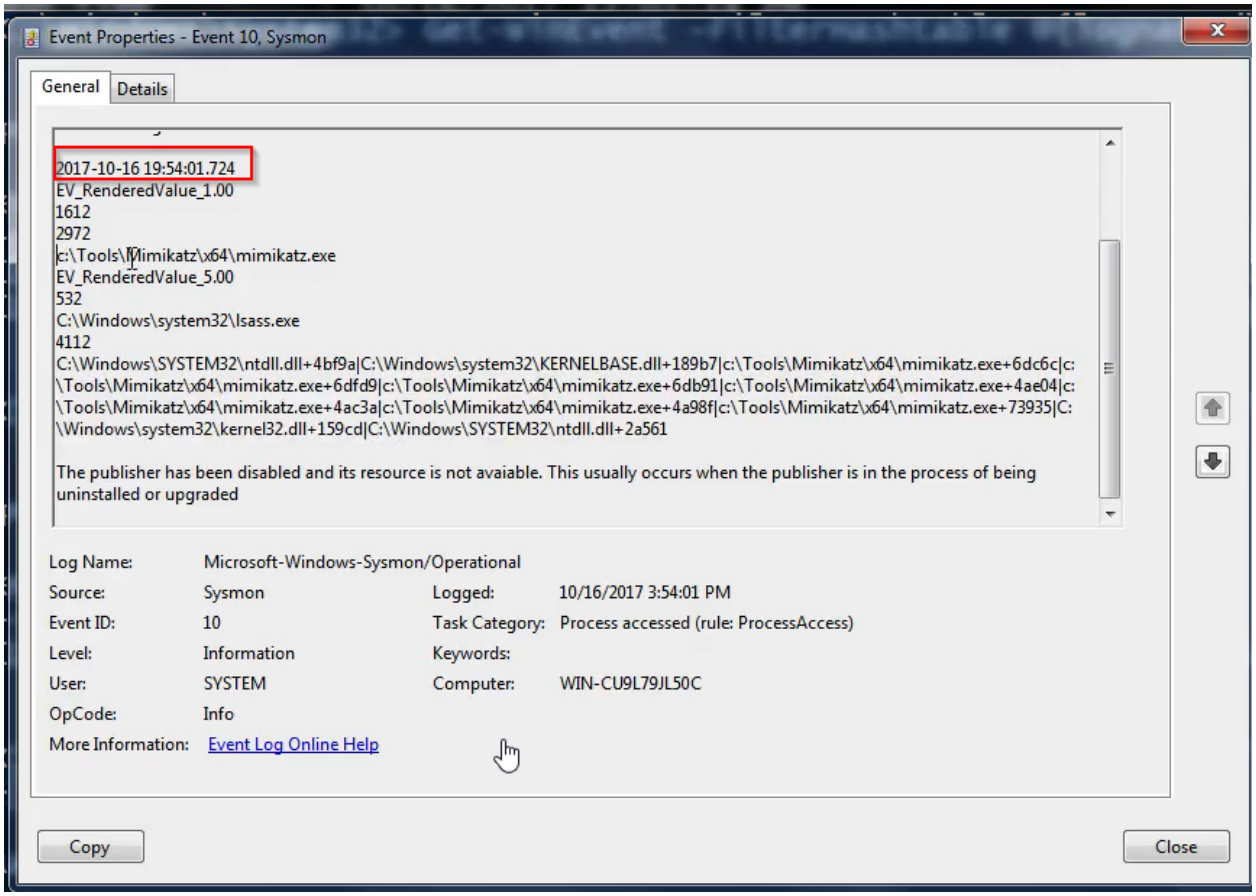
```
PS C:\Windows\system32> Get-WinEvent -FilterHashtable @{logname="Microsoft-Windows-Sysmon/Operational";id=10}


   ProviderName: Microsoft-Windows-Sysmon

TimeCreated                     Id LevelDisplayName Message
-----------                     -- ---------------- -------
10/16/2017 3:54:01 PM           10 Information      Process accessed:...
```

Here we add the ;id=10 which is filtering for Event ID 10, so now we see the events when a process accessed another process.

Lets get the properties of the log as we did by double clicking in the Event Viewer, we want the same details here:

```
PS C:\Windows\system32> Get-WinEvent -FilterHashtable @{logname="Microsoft-Windows-Sysmon/Operational";id=10}
| %{$_.Properties[0]}

Value
-----
2017-10-16 19:54:01.724
```

Here we added "| %{$_.Propertoes[0]}" here we wanted to grab the properties but we want the values in index 0, mean the first thing which is the time as we also saw in the event viewer we saw the time:



As we see the first thing is the time so with 0 we access it, but to get the source image which is the mimkatz.exe, for that we need to put 4 as it's in the 4 line mean it's the 4th thing. So we will first go to the Event viewer to get the idea of where we can find what info and then use these IDIAs in filtering in powershell

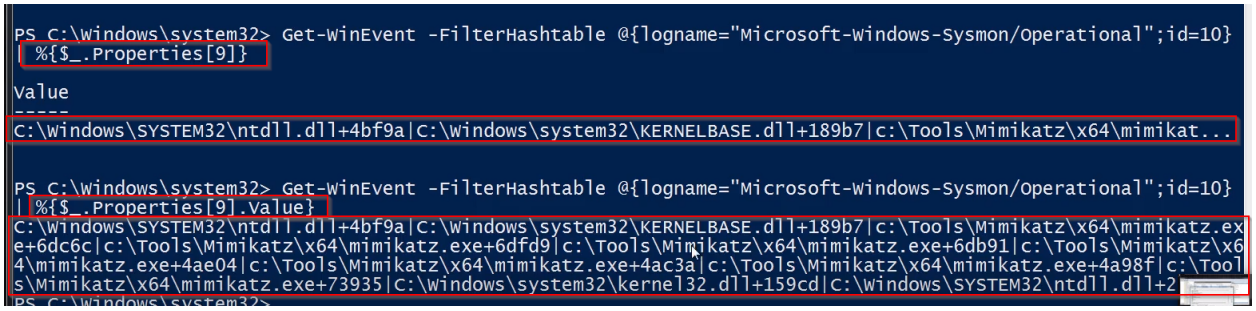
```
PS C:\Windows\system32> Get-WinEvent -FilterHashtable @{logname="Microsoft-Windows-Sysmon/Operational";id=10}
| %{$_.Properties[4]}

Value
-----
c:\Tools\Mimikatz\x64\mimikatz.exe
```

Here we put 4, and we got the image that did the accessing which is the mimkatz.exe.

We can do the same for the target image, mean the image that was targeted to be access which is lsass.exe and its index location is 7:

```
PS C:\Windows\system32> Get-WinEvent -FilterHashtable @{logname="Microsoft-Windows-Sysmon/Operational";id=10}
| %{$_.Properties[7]}

Value
-----
C:\Windows\system32\lsass.exe
```

But if we access the 9th index location, we will not get all the details of it and to do so we will add .value to our properties filter:


PS C:\Windows\system32> Get-WinEvent -FilterHashtable @{logname="Microsoft-Windows-Sysmon/Operational";id=10}
| %{$_.Properties[9]}
Value

C:\Windows\SYSTEM32\ntdll.dll+4bf9a|C:\Windows\system32\KERNELBASE.dll+189b7|c:\Tools\Mimikatz\x64\mimikat...
PS C:\Windows\system32> Get-WinEvent -FilterHashtable @{logname="Microsoft-Windows-Sysmon/Operational";id=10}
| %{$_.Properties[9].Value}
C:\Windows\SYSTEM32\ntdll.dll+4bf9a|C:\Windows\system32\KERNELBASE.dll+189b7|c:\Tools\Mimikatz\x64\mimikatz.ex
e+6dc6c|c:\Tools\Mimikatz\x64\mimikatz.exe+6dfd9|c:\Tools\Mimikatz\x64\mimikatz.exe+6db91|c:\Tools\Mimikatz\x6
4\mimikatz.exe+4ae04|c:\Tools\Mimikatz\x64\mimikatz.exe+4ac3a|c:\Tools\Mimikatz\x64\mimikatz.exe+4a98f|c:\Tool
s\Mimikatz\x64\mimikatz.exe+73935|C:\Windows\system32\kernel32.dll+159cd|C:\Windows\SYSTEM32\ntdll.dll+2
PS C:\Windows\system32>